(19) JAPANESE PATENT OFFICE (JP)
(12) Gazette of Unexamined Patent Applications (A)
(11) Unexamined Patent Application (Kokai) No.
H11-328318

(43) Publication Date: November 30 1999

(51) Int. Cl.[6]:	ID Code:	FI	
G06K 9/72		G06K 9/72	C
			Z

Request for Examination: Not requested
No. of Claims: 17
OL (Total of 21 pages)

(21) Patent Application No.: H10-127938

(22) Filing Date: May 11 1998

Pursuant to the provisions of the proviso to paragraph 2, Article 64 of the Patent Law, a passage marked by the symbol "x" has not been published

(71) Applicant: 000005049
Sharp Corp.
22-22 Nagaike-cho, Ageno,
Osaka-shi, Osaka
(72) Inventor: Hideaki Tanaka
c/o Sharp Corp.
22-22 Nagaike-cho, Ageno,
Osaka-shi, Osaka
(74) Agent: Patent Attorney, Shigeru Aoyama
(and 1 other)

(54) [Title of the Invention] Probability table generating apparatus, probability system language processing apparatus, recognition apparatus, and recording medium

(57) [Abstract]
[Problem] To facilitate high-speed, small memory capacity high-accuracy probability system language processing.
[Solving Means] A unigram generator 19 obtains a unigram. A digram generator 20 obtains a digram. An attribute trigram generator 21 obtains an attribute trigram. A vector divider 23 divides the digram into vectors of recognition object character number dimension. A clusterer 24 clusters the vectors to generate a compressed character code conversion table 13 of characters having correlation with cluster codes (compressed character codes). A compressed digram generator obtains a transition probability (compressed digram) of 2-compressed character code sets. The number of elements in the compressed digram is highly compressed without loss of the linguistic transition information of the digram. In addition, lost linguistic information is compensated by use of the unigram and the attribute trigram in combination which, in turn, facilitates a high-accuracy language processing without loss of the small memory capacity and high-speed characteristics.

1 DOCUMENT DATABASE

2 UNIGRAM BUFFER

3 TOTAL CHARACTER NUMBER COUNTER

4 DIGRAM BUFFER

5 TOTAL 2-CHARACTER COMBINATION NUMBER COUNTER

6 ATTRIBUTE TRIGRAM BUFFER

7 TOTAL 3-CHARACTER COMBINATION NUMBER COUNTER

8 COMPRESSED DIGRAM BUFFER

9 TOTAL 2-COMPRESSED CHARACTER CODE COMBINATION NUMBER COUNTER

10 VECTOR BUFFER

11 CLUSTER BUFFER

12 CHARACTER ATTRIBUTE CONVERSION TABLE

13 COMPRESSED CHARACTER CODE CONVERSION TABLE

14 TEMPORARY MEMORY

15 CURRENT CHARACTER BUFFER

16 PREVIOUS CHARACTER BUFFER

17 PRE-PREVIOUS CHARACTER BUFFER

18 CONTROLLER

19 UNIGRAM GENERATOR

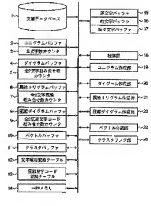
20 DIGRAM GENERATOR

21 ATTRIBUTE TRIGRAM GENERATOR

22 COMPRESSED DIGRAM GENERATOR

23 VECTOR DIVIDER

24 CLUSTERER



[Claims]

[Claim 1] A probability table generating apparatus comprising:

a memory in which one natural language character string is stored;

a clusterer for clustering all characters of the character string stored in said memory into clusters having similar transition characteristics and assigning a cluster code to the clusters; and

a 2-cluster code transition probability table generator for generating a 2-cluster code transition probability table based on obtaining cluster codes of the characters of the character string stored in said memory and, for all adjacent 2-characters in said character string, obtaining a transition probability between the cluster codes of these 2-characters.

[Claim 2] The probability table generating apparatus as claimed in claim 1, further comprising:

a 3-character attribute transition probability table generator for generating a 3-character attribute transition probability table based on obtaining attributes of the characters of the character string stored in said memory and, for all adjacent 3-characters in said character string, obtaining a transition probability between the attributes of these 3-characters.

[Claim 3] The probability table generating apparatus as claimed in claim 2, further comprising:

a 1-character appearance transition probability table generator for generating a 1-character appearance transition probability table based on, for all characters in said character string stored in said memory, obtaining an appearance probability of these 1-characters.

[Claim 4] The probability table generating apparatus as claimed in claim 1, further comprising:

a logarithm compressor for logarithmically compressing element values of said 2-cluster code transition probability table.

[Claim 5] The probability table generating apparatus as claimed in claim 1, further comprising:
a 2-character transition probability table generator for generating a 2-character transition probability table based on obtaining a transition probability between all adjacent 2-characters of all characters of a character string stored in said memory,
wherein said clusterer, which clusters all vectors produced by division of said 2-character transition probability table into recognition object character number vectors of a recognition object character number dimension into a predetermined number of clusters, comprises a cluster code conversion table for, in accordance with a result of said clustering, generating a cluster code conversion table employed for obtaining cluster codes of the clusters to which each said 2-characters belong, and
said 2-cluster code transition probability table generator, which comprises a cluster code converter for converting each of said 2-characters to said cluster codes employing said cluster code conversion table, employs the converted cluster codes to generate said 2-cluster code transition probability table.

[Claim 6] The probability table generating apparatus as claimed in claim 2, further comprising:
an attribute conversion table employed for obtaining the attributes of said characters,
wherein said 3-character attribute transition probability table generator, which comprises an attribute converter for converting said characters to attributes employing said attribute conversion table, employs said converted attributes to generate said 3-character attribute transition probability table.

[Claim 7] The probability table generating apparatus as claimed in claim 2,
wherein a Japanese language document is stored in said memory,
said characters are Japanese language characters, and
said attributes are any of hiragana, katakana, symbols, kanji, numerals, alphabet upper case and alphabet lower case characters.

[Claim 8] The probability table generating apparatus as claimed in claim 2,
wherein a Chinese language document is stored in said memory,
said characters are Chinese language characters,
and said attributes are any of joji, symbols, kanji other than joji, numerals, alphabet upper case and alphabet lower case characters.

[Claim 9] A probability system language processing apparatus comprising:
said 2-cluster code transition probability table generated by the probability table generating apparatus as claimed in claim 1; and
a character string transition probability calculator for, employing said 2-cluster code transition probability table, calculating a character string transition probability of an input character string.

[Claim 10] The probability system language processing apparatus as claimed in claim 9, comprising:
at least one of said 1-character appearance probability table and the 3-character attribute transition probability table generated by the probability table generating apparatus as claimed in claim 2 and claim 3,
wherein said character string transition probability calculator also employs said probability tables to calculate the character string transition probability of said input character string.

[Claim 11] The probability system language processing apparatus as claimed in claim 10,

wherein said character string transition probability calculator calculates said character string transition probability in accordance with a following 1st equation:

$P(c_1, c_2, \ldots, c_n) = P(c_1) + P(c_1, c_2) + P(c_1, c_2, c_3) + P(c_2, c_3, c_4) + \ldots + P(c_{n-2}, c_{n-1}, c_n) + P(c_{n-1}, c_n) + P(c_n)$

Here,

P(x) = 1-character appearance probability table (x)

P(x, y) = {1-character appearance probability table (x) + 1-character appearance probability table (y)}/2+2-cluster code transition probability table (x^c, y^c)/2

P(x, y, z) = {1-character appearance probability table (x) + 1-character appearance probability table (y) + 1-character appearance probability table (z)}/3+{2-cluster code transition probability table (x^c, y^c) + 2-cluster code transition probability table (y^c, z^c)}/2 + {3-character attribute transition probability table (x^a, y^a, z^a)}/3

and,

x, y, z : said characters

x^c, y^c, z^c : said cluster codes of characters x, y, z

x^a, y^a, z^a : attributes of characters x, y, z

1-character appearance probability table (x) : element values of language element x of 1-character appearance probability table

2-cluster code transition probability table (x^c, y^c): element values of cluster code sets (x^c, y^c) of 2-cluster code transition probability table

3-character attribute transition probability table (x^a, y^a, z^a): element values of character attribute sets (x^a, y^a, z^a) of 3-character attribute transition probability table

[Claim 12] A recognition apparatus, comprising: a segmenter for segmenting individual language elements from an input language element string; a matcher for matching feature patterns of the segmented language

elements with a standard pattern to produce a plurality of recognition candidates; and a candidate character string generator for, by implementing a probability system language processing on the plurality of candidate character strings produced by combining said plurality of recognition candidates, generating a predetermined number of candidate character strings,
and further comprising said 2-cluster code transition probability table generated by the probability table generating apparatus as claimed in claim 1,
wherein said candidate character string generator, which comprises a score calculator for calculating a score of said candidate character string employing said 2-cluster code transition probability table, implements a probability system language processing for estimating a likelihood of said candidate character string in accordance with said calculated score.

[Claim 13] The recognition apparatus as claimed in claim 12, further comprising:
at least one of said 1-character appearance probability table and the 3-character attribute transition probability table generated by the probability table generating apparatus as claimed in claim 2 and claim 3,
wherein said score calculator also employs said probability tables to calculate said candidate character string score.

[Claim 14] The recognition apparatus as claimed in claim 12, further comprising:
a word inquiry language processor for, by implementing a word inquiry system language processing on a predetermined number of candidate character strings generated by said candidate character string generator, outputting an optimum candidate character string as a recognition result of said input language element string.

[Claim 15] The recognition apparatus as claimed in

claim 13,
wherein said score calculator calculates said score in accordance with the following 2nd equation:
Score (1) = $W_S S_1 + W_F P(c_1)$
Score (2) = $W_S\{(S_1 + S_2)/2\} + W_F P(c_1, c_2)$
Score (i) = [Score (i-1) x (i-1) + $\{W_S S_i + W_F P(c_{i-2}, C_{i-2}, c_i)\}$]/1
Here,
Score (I) : Score of candidate character strings of character number I
$c_1, c_2, c_3, \ldots, c_n$: Said candidate character strings
$S_1, S_2, S_3, \ldots, S_n$: Said matching estimated values of said recognition candidates
P(y) : Character string transition probability of candidate character string y (calculation performed employing the Equation P(x), P(x, y), P(x, y, z) described in claim 7)
W_F, W_S : Weightings

[Claim 16] The recognition apparatus as claimed in claim 12,
wherein said input language element string is a character string.

[Claim 17] A computer-readable recording medium on which is recorded a character recognition program for making a computer function as:
a segmenter that segments individual characters from an input character string, a matcher that produces a plurality of recognition candidates based on matching feature patterns of segmented characters with a standard pattern,
and a candidate character string generator that, by implementing a probability system language processing in which said 2-cluster code transition probability table generated by the probability table generating apparatus described in claim 1 is employed on the plurality of candidate character strings produced by

combining said plurality of recognition candidates, calculates a score of said candidate character strings and, in accordance with this score, generates a predetermined number of candidate strings.

[Detailed Description of the Invention]
[0001]
[Field of Technology to which the Invention Belongs] The present invention relates to a probability table generating apparatus for generating probability tables used for small capacity memory, high-speed, high-recognition rate language processing of text or speech, a probability system language processing apparatus and recognition apparatus employing these probability tables, and a computer-readable recording medium.
[0002]
[Prior Art] Developments in recognition techniques in recent years have led to the widespread popularization amongst users of high-accuracy recognition apparatuses and recognition software of a performance level that is suitable for practical application, as well as commercial products in which these apparatuses and software have application. One example thereof is a portable information terminal (PDA: personal digital assistant) of which an essential function is handwritten character recognition.
[0003] While the main factors behind the improvement in recognition accuracy in recent years obviously include the tuning, reforming and devising of methods for improving text and speech unit recognition accuracy, the progress that has been achieved in so-called language processing techniques (also referred to as knowledge processing) may be thought of as another significant factor.
[0004] The importance and necessity of language processing techniques in text and speech recognition apparatuses is reflected in the following two reasons:
(1) First level recognition rate using a recognition technique unit remains low. Even using current highly

accurate recognition methods, the first level recognition rate of the character unit of, for example, a Japanese language typeface OCR (optical character reader) is a low 97% to 98%. However, the candidate recognition rate at around the 10^{th} level exceeds 99%, and the practical application of a recognition apparatus of even higher accuracy necessitates the implementation of a method of some kind for selecting a correct candidate. There is even greater necessity for a language processing technique for handwritten characters or speech in which the recognition rate is lower than for printed text.

[0005] (2) Accurate segmentation is impossible using recognition estimated values alone. The most commonly used method in existing recognition apparatuses is based on a combination of segmentation and recognition processing. However, there is a problem inherent even to this method in that, because of the unstable performance of recognition techniques as described above, a drop in segmentation accuracy occurs in segmentation in which there is a reliance on recognition estimated values (distance and similarity and so on). Furthermore, there are some patterns that simple cannot be segmented using the abovementioned recognition estimated values alone. Taking the Japanese character "加 " as an example, it is impossible to distinguish by pattern alone whether this character is configured from the half-width hiragana characters "力" and "口" or a full-width kanji character "加 ".

[0006] The importance and need for language processing based on the reasons (1), (2) described above has been recognized from the beginnings of character recognition research, and a range of modifications over time have led to the development of the currently available language processing techniques. Language processing is now an essential component of all types of recognition apparatuses.

[0007] Most language processing trials carried out in Japan since the beginning of the development of

recognition apparatuses have focussed on word inquiry. This language processing involves judgment by means of a word dictionary search as to whether or not a correct recognition result is contained in a character string produced by combining recognition candidates. This kind of contextual processing has proved a highly effective language processing means (its use together with recognition processing including segmentation and recognition has been considered), and has been conventionally adopted in a wide range of recognition apparatuses.

[0008] However, there are problems in the language processing fundamentally based on word inquiry described above in that: (1) character strings not registered in the word dictionary cannot be accurately recognized; and (2) when there is an absence of demarcation between the words in a sentence which is the case in, in particular, the Japanese language, accurate segmentation (morpheme analysis) of the space in which the word inquiry is being performed is impossible, and the probability system language processings disclosed in [1] Japanese Unexamined Patent Application No. H5-6464 (method and apparatus for character string recognition) and [2] Japanese Unexamined Patent Application No. H5-120494 as (method and apparatus for character string recognition) have been devised as measures to overcome these problems.

[0009] The probability system language processings described in, for example, Japanese Unexamined Patent Applications [1] and [2] are fundamentally based on a hypothesizing of the character string as a Markov model, and while the Unexamined Patent Application [1] discloses the use of a binary digram (term that expresses the appearance frequency or probability of a 2-character combination), the Unexamined Patent Application [2] discloses the use of a multiple value modified digram.

[0010] The probability system language processings described above have been hitherto executed using US

and European technology. This has been reported in, for example, [3] E. Riseman and A. Hanson “A contextual postprocessing system for error correction using binary n-grams,) IEETrans, Comput., pp480-493, May 1974. and [4] A. Goshtasby, Contextual word recognition using probabilistic relaxation labelling,), Pattern Recognition, vol. 21, no.5. pp. 455-462 1988. Notably, cited reference [3] describes the use of 2-value digram and cited reference [4] describes the use of a multiple value digram.

[0011] Similarly to the research conducted in Japan, the early-stage language processing research carried out in Europe and the US focussed on the development of a spell-check (word inquiry) system. However, the focus of the research subsequently shifted to the development of a probability system for resolving the problem of non-registration described above, and probability systems developed as an outcome thereof are currently in widespread use (although probability systems used in conjunction with spell-check systems are currently widely employed).

[0012] This shift in Europe and the US towards the development of a probability system occurred despite the fact that the problem of word segmentation (morpheme analysis) is comparatively simpler to resolve than in Japan. This reflects the large number of non-registered words that must be input into a recognition apparatus for even general use and, until this problem is unresolved, the development of recognition apparatus suitable for all practical applications will remain unrealized. The merits and problems inherent to the probability system language processing described above are outlined below.

[0013] (A) Merits of probability system language processing

A probability system evaluates the likelihood of a character string without need for morpheme segmentation. For example, for the “日 “ of “本日は 晴天なり“, a recognition result of either “目 “ or “日 “ is

produced, and the two character strings “本日は 晴天 なり“ and “本目は 晴天 なり“ are generated. If the probability value of these character strings is P(本日は 晴天 なり) = 0.71 and P(本日 は 晴天 なり) = 0.76 respectively, the language processing system will correctly select the recognition result “目” . The merits of a language processing based on this probability system are listed below.

· A correct answer for words not registered in the dictionary such as peculiar nouns can (possibly) be obtained

· Word segmentation is unnecessary. The language processing is effective for language in which the morpheme analysis is imperfect such as, for example, Japanese.

· The language processing functions using indefinite morphemes (for example, during input) and, accordingly, language processing is possible at an incomplete stage of input when using a recognition apparatus of a type in which a point-by-point input occurs such as for the handwriting recognition of a PDA

· The language processing is fast

[0014] A method of transition probability calculation using an actual character string will be hereinafter described. According to the cited reference “Speech recognition using probability models) (Seiichi Nakagawa, The Institute of Electronics, Information and Communication, Corona Co., Ltd, First published 1988), taking the character string as C = (c_1, c_2, c_3, …, c_n) and F() as the appearance frequency of the combination of the characters within the (), the transition probability P(C) of the character string C is expressed by Equation (1).

[Equation 1]

$$P(C)=\prod_{i=1}^{n}\frac{F(c_{i-2},c_{i-1},c_i)}{F(c_{i-2},c_{i-1})} \qquad \cdots (1)$$

[0015] Subsequent to the preparation of 2-character set (digram) and 3 character set (trigram) appearance frequency (or probability) tables, the transition probability P(C) of the arbitrary length character string C can be calculated from Equation (1). In addition, it is self-evident from Equation (1) that the transition probability value can be calculated without need for segmentation of individual words and that, accordingly, morpheme analysis is unnecessary. In addition, because the abovementioned transition probability value constitutes a probability value (estimated value of contextual likelihood), even if the word is unregistered, it increases according to the linguistic likelihood of the word. Accordingly, the correct recognition candidate can be selected in accordance with the transition probability value.

[0016] Furthermore, while a multiplication/division calculation is employed in the abovementioned Equation (1), the digram and trigram are generally expressed as transition probabilities and are converted to the addition/subtract calculation of Equation (2) by logarithmic conversion (compression). The transition probability P(C) of the character string C is calculated by the addition/subtraction calculation of Equation (2). As a result, the transition probability P(C) can be calculated by a very simple processing based on reference to the digram and trigram transition probability tables and an addition/subtraction calculation and, accordingly, the processing can be implemented much faster than is possible with language processing based on a dictionary search (word inquiry).

[Equation 2]

$$\log\{P(C)\} = \sum_{i=1}^{n} [\log\{P(c_{i-2}, c_{i-1}, c_i)\} - \log\{P(c_{i-2}, c_{i-1})\}] \quad \cdots(2)$$

[0017] (B) Problems inherent to probability system language processing

Problems in language processing based on the use of a probability system are outlined below.

· Because the transition probability value constitutes a (multiple value) probability value, a definite estimate value is difficult to determine. With language processing based on word inquiry the processing result comprises two values “yes”/”no” and, accordingly, the processing result is easily determined. In contrast, multiple transition probability values make the determining of a definite estimate value of the processing result difficult. More particularly, if a recognition apparatus has a fully complete vocabulary then word inquiry is more effective for determining the processing result.

· When the recognition object character number is large the memory capacity is huge

[0018] The first of the problems of the probability system language processing described above can be avoided by combined use with word inquiry system language processing. This method is commonplace in the US and Europe. The second of the problems described above pertaining to memory capacity is particularly evident in the case of the Japanese language.

[0019] With regard to the memory capacity of the digram transition probability table and the trigram transition probability table described above, while on the one hand the memory capacity of the digram transition probability table increases proportionately to the square of the appearance number, in other words, the recognition object character number, of 2-charcter combinations, the memory capacity of the trigram transition probability table increases proportionally to the cube of the appearance number, in other words the recognition object character number, of 3-character combinations. Using the Japanese language typeface OCR as an example, as shown in FIG. 9, the memory capacity where the recognition object is taken to be 4000 characters and one element of the transition probability table is taken to be 1 byte is 16 megabytes for a digram and 64 megabytes for a trigram which are unrealizable memory capacities. Notably, the

probability tables themselves described above are hereinafter referred to as a digram and trigram.
[0020] The simplest means for resolving the second of the problems is to calculate the transition probability P(C) using only a digram while without using the trigram (a natural outcome thereof is a drop in the accuracy of the transition probability P(C) obtained). A significant memory capacity is nevertheless required. While an expensive, small memory capacity, probability system language processing for this purpose was able to be realized as early as 10 to 20 years ago for European languages and English in which the number of characters is smaller, the realization of this language processing for the Japanese language has proved difficult. Subsequent to advancements in terms of cost reduction and increased size of memory achieved in recent years, the use of probability system language processing has now begun in Japan as well.
[0021] In the Japanese Unexamined Patent Applications of [1] and [2] described above, the problem of memory capacity is dealt with by the use of a digram alone without using a trigram. Furthermore, in the Unexamined Patent Application of [1], the memory capacity is further reduced using a 2-value digram without using a multiple value digram. In addition, in the Unexamined Patent Application of [2], the memory capacity is further reduced by the probability values of combinations of a fixed probability value or above alone being stored.
[0022]
[Problems to be Solved by the Invention] However, the following problems are inherent to the conventional probability system language processings disclosed in the Unexamined Patent Applications [1] and [2] described above. That is to say, there are problems in the probability system language processing of Unexamined Patent Application [1] in that, because a 2-value digram is used, the accuracy of the obtained transition probability P(C) is significantly lower than

when a multiple value digram is used. Furthermore, while the memory capacity is reduced by use of a 2-value digram, the memory capacity is still a large 2 megabytes (=4000 x 4000/8). More particularly, combined use with a word inquiry system language processing necessitates a separate capacity for a language processing dictionary and, accordingly, the memory capacity referred to above must as far as possible be reduced.

[0023] In addition, in the probability system language processing of Unexamined Patent Application [2], because the probability value of only combinations of a first probability value or above are stored in the transition probability table, when the transition probability P(C) of a certain 2-character combination is to be obtained, a search must always be carried out to determined whether the transition probability of this 2-character combination is stored in the transition probability table. Accordingly, there is a problem from the viewpoint of the loss of high-speed characteristics.

[0024] Thereupon, it is an object of the invention to provide a probability table generating apparatus for generating a multiple-value probability table in which a high transition probability is produced without loss of high-speed characteristics and small memory capacity is possible, and a probability system language processing apparatus and recognition apparatus in which the probability table is employed.

[0025]

[Means to Solve the Problems] The probability table generating apparatus pertaining to claim 1 for achieving this object comprises a memory in which one natural language character string is stored, a clusterer for clustering all characters of the character string stored in the abovementioned memory into clusters having similar transition characteristics and assigning a cluster code to the clusters, and a 2-cluster code transition probability table generator for

generating a 2-cluster code transition probability table based on obtaining cluster codes of the characters of the character string stored in the abovementioned memory and, for all adjacent 2-characters in the abovementioned character string, obtaining a transition probability between the cluster codes of these 2-characters.

[0026] According to the abovementioned configuration, the character or syllable number of a recognition object is compressed to the abovementioned cluster number. Accordingly, the 2-cluster code transition probability table that expresses the transition probability between cluster codes of adjacent 2-characters facilitates a small memory capacity, high-speed probability language processing in which element values are compressed proportionately to the 2-character transition probability table that expresses the transition probability between adjacent 2-characters to (cluster number/recognition object character (or syllable) number)^2. Moreover, because the characters or syllables belonging to the same cluster comprise the same transition characteristics, the linguistic transition information of the characters or syllables is retained in the above-noted 2-cluster code transition probability table. Accordingly, a high accuracy probability system language processing can be implemented employing the abovementioned multiple value 2-cluster code transition probability table.

[0027] In addition, the invention pertaining to claim 2 of the probability table generating apparatus of the invention pertaining to claim 1 further comprises a 3-character attribute transition probability table generator for generating a 3-character attribute transition probability table based on obtaining attributes of the characters of the character string stored in the abovementioned memory and, for all adjacent 3-characters in said character string, obtaining a transition probability between the attributes of these 3-characters.

[0028] According to the abovementioned configuration, a 3-character attribute transition probability table that expresses the transition probability between adjacent 3-characters in natural language is generated. Accordingly, by implementing a probability system language processing based on combined use of a 3-character attribute transition probability table and the abovementioned 2-cluster code transition probability table, the character or syllable linguistic transition information lost when the total number of characters and syllables of a recognition object are compressed can be compensated and, as a result, the accuracy of the probability system language processing can be further improved.

[0029] In addition, the invention pertaining to claim 3 of the probability table generating apparatus of the invention pertaining to claim 2 further comprises a 1-character appearance transition probability table generator for generating a 1-character appearance transition probability table based on, for all characters in the abovementioned character string stored in the abovementioned memory, obtaining an appearance probability of these 1-characters.

[0030] According to the abovementioned configuration, a 1-character appearance probability table that expresses the appearance probability of characters in natural language is generated. Accordingly, by implementing a probability system language processing based on combined use of the abovementioned 1-character appearance probability table and the 3-character attribute transition probability table with the abovementioned 2-cluster code transition probability table, the character and syllable linguistic transition information lost when the total number of character and syllables of a recognition object are compressed can be compensated and, as a result, the accuracy of the probability system language processing can be further improved.

[0031] In addition, the invention pertaining to claim 4

of the probability table generating apparatus of the invention pertaining to claim 1 comprises a logarithm compressor for logarithmically compressing element values of the abovementioned 2-cluster code transition probability table.

[0032] According to the abovementioned configuration, the element values of the abovementioned 2-cluster code transition probability table are compressed so as to be representable using 1 byte. In this way, a further reduction in memory capacity can be achieved and, accompanying this, the speed of the probability system language processing can be further increased.

[0033] In addition, the invention pertaining to claim 5 of the probability table generating apparatus of the invention pertaining to claim 1 comprises a 2-character transition probability table generator for generating a 2-character transition probability table based on obtaining a transition probability between all adjacent 2-characters of all characters of a character string stored in the abovementioned memory, wherein the abovementioned clusterer, which clusters all vectors produced by division of the abovementioned 2-character transition probability table into recognition object character number vectors of a recognition object character number dimension into a predetermined number of clusters, comprises a cluster code conversion table for, in accordance with a result of the abovementioned clustering, generating a cluster code conversion table employed for obtaining cluster codes of the clusters to which each the abovementioned 2-characters belong, and the abovementioned 2-cluster code transition probability table generator, which comprises a cluster code converter for converting each of the abovementioned 2-characters to the abovementioned cluster codes employing the abovementioned cluster code conversion table, employs the converted cluster codes to generate the abovementioned 2-cluster code transition probability table.

[0034] According to the abovementioned configuration, a

2-character transition probability table that expresses the transition probability between adjacent 2-characters in the abovementioned character string is generated. In addition, characters or syllables of similar transition characteristics are clustered in the same cluster in accordance with the 2-character transition probability table. Accordingly, by assigning cluster codes to the clusters and reckoning the cluster codes as characters, the total character or syllable number of a recognition object is compressed to the abovementioned cluster number without loss of the linguistic transition information of the characters and syllables. In addition, a cluster code conversion table employed for obtaining the abovementioned cluster codes is generated. The cluster codes for the abovementioned adjacent 2-characters are produced and the abovementioned 2-cluster code transition probability table generated in this way by a simple processing involving reference to the abovementioned cluster code conversion table.

[0035] In addition, the invention pertaining to claim 6 of the probability table generating apparatus of the invention pertaining to claim 2 further comprises an attribute conversion table employed for obtaining the attributes of said characters, wherein said 3-character attribute transition probability table generator, which comprises an attribute converter for converting said characters to attributes employing said attribute conversion table, employs said converted attributes to generate said 3-character attribute transition probability table.

[0036] According to the abovementioned configuration, the abovementioned 3-character attribute transition probability table is generated by a simple processing involving reference to an attribute conversion table.

[0037] According to the invention pertaining to claim 7 of the probability table generating apparatus pertaining to claim 2, a Japanese language document is stored in the abovementioned memory, the abovementioned

characters are Japanese language characters, and the abovementioned attributes are any of hiragana, katakana, symbols, kanji, numerals, alphabet upper case and alphabet lower case characters.

[0038] According to the abovementioned configuration, in order to attain a recognition object character number of the order of 4000, the memory capacity of a Japanese language character 2-character transition probability table for which, taking 1 element as 1 byte, 16 megabytes is required, is compressed to 1 megabyte by generating the abovementioned 2-cluster code transition probability table based on character compression in which, for example, the abovementioned cluster number is clustered to 1000. Similarly, the storage capacity of the 3-character transition probability table for which 64 megabytes is required is compressed to a maximum 343 bytes by generating the abovementioned 3-character attribute transition probability table based on conversion of the recognition object characters to any of the attributes of hiragana, katakana, symbols, kanji, numerals, alphabet upper case and alphabet lower case characters. A small memory capacity probability table loadable in a Japanese language character recognition apparatus that expresses the transition information between adjacent 2-characters and adjacent 3-characters is generated in this way.

[0039] In addition, in the invention pertaining to claim 8 of the probability table generating apparatus as claimed in claim 2, a Chinese language document is stored in the abovementioned memory, the abovementioned characters are Chinese language characters, and the abovementioned attributes are any of joji, symbols, kanji other than joji, numerals, alphabet upper case and alphabet lower case characters.

[0040] According to the abovementioned configuration, in the case of Chinese language in which, similarly to Japanese language, the number of recognition object characters is very large, the memory capacity of the 2-

character transition probability table is compressed to, for example, 1 megabyte, by generating the abovementioned 2-cluster code transition probability table by character number compression based on clustering of the recognition object characters. Similarly, the storage capacity of the 3-character attribute transition probability table is compressed to, for example, a maximum of 216 bytes by generating the abovementioned 3-character attribute transition probability table based on conversion of the recognition object characters to any of the attributes of joji, symbols, kanji other than joji, numerals, alphabet upper case and alphabet lower case characters.

[0041] In addition, the probability system language processing apparatus pertaining to claim 9 comprises the abovementioned 2-cluster code transition probability table generated by the probability table generating apparatus as claimed in claim 1, and a character string transition probability calculator for, employing the abovementioned 2-cluster code transition probability table, calculating a character string transition probability of an input character string.

[0042] According to the abovementioned configuration, the memory capacity of the abovementioned 2-cluster code transition probability table used in the probability system language processing is compressed by an amount (cluster number/recognition object character (or syllable) number)^2 more than a conventional 2-character transition probability table. For this reason, even if the processing object constitutes Japanese language characters of recognition object character number of the order of 4000, taking 1 element as 1 byte, a 1-megabyte memory capacity is sufficient.

[0043] The invention pertaining to claim 10 of the probability system language processing apparatus as claimed in claim 9 comprises at least one of the abovementioned 1-character appearance probability table and the 3-character attribute transition probability table generated by the probability table generating

apparatus as claimed in claim 2 and claim 3, the abovementioned character string transition probability calculator also employs the abovementioned probability tables to calculate the character string transition probability of the abovementioned input character string.

[0044] According to the abovementioned configuration, high-accuracy character string transition probability is calculated in the abovementioned probability system language processing by the combined use of at least one of the abovementioned 1-character appearance probability table and the 3-character attribute transition probability table which constitutes a table that expresses the transition information between adjacent 3-characters.

[0045] In addition, in the invention pertaining to claim 11 of the probability system language processing apparatus as claimed in claim 10, the abovementioned character string transition probability calculator calculates the abovementioned character string transition probability in accordance with the abovementioned 1^{st} equation indicated above.

[0046] According to the abovementioned configuration, the abovementioned character string transition probability is calculated by a very simple calculation processing involving reference to the abovementioned 2-cluster code transition probability table, the 3-character attribute transition probability table and the 1-character appearance probability table. A further increase in the speed of probability system language processing actuation is achieved in this way.

[0047] In addition, the recognition apparatus of the invention pertaining to claim 12 comprises a segmenter for segmenting individual language elements from an input language element string, a matcher for matching feature patterns of the segmented language elements with a standard pattern to produce a plurality of recognition candidates, and a candidate character string generator for, by implementing a probability

system language processing on the plurality of candidate character strings produced by combining the abovementioned plurality of recognition candidates, generating a predetermined number of candidate character strings, and further comprises the abovementioned 2-cluster code transition probability table generated by the probability table generating apparatus as claimed in claim 1, wherein the abovementioned candidate character string generator, which comprises a score calculator for calculating a score of the abovementioned candidate character string employing the abovementioned 2-cluster code transition probability table, implements a probability system language processing for estimating a likelihood of the abovementioned candidate character string in accordance with the abovementioned calculated score.

[0048] According to the abovementioned configuration, the score used for estimating the likelihood of a candidate character string in the abovementioned probability system language processing is calculated using the abovementioned 2-cluster code transition probability table in which the element number is compressed (cluster number/recognition object character (or syllable) number)^2 more than a conventional 2-character transition probability table. For this reason, even when Japanese language characters of recognition object character number of the order of 4000 are to be recognized, taking 1 element as 1 byte, a memory capacity of 1 megabyte is sufficient, and there are no impediments at all to the loading of a word inquiry-system language processing dictionary.

[0049] In addition, the invention pertaining to claim 13 of the recognition apparatus of the invention pertaining to claim 12 further comprises at least one of the abovementioned 1-character appearance probability table and the 3-character attribute transition probability table generated by the probability table generating apparatus as claimed in claim 2 and claim 3, wherein the abovementioned score

calculator also employs the abovementioned probability tables to calculate the abovementioned candidate character string score.

[0050] According to the abovementioned configuration, a score that more accurately expresses the likelihood of the abovementioned candidate character string can be calculated by the combined use of at least one of the abovementioned 1-character appearance probability table and the 3-character attribute transition probability table.

[0051] The invention pertaining to claim 14 of the recognition apparatus pertaining to claim 12 further comprises a word inquiry language processor for, by implementing a word inquiry system language processing on a predetermined number of candidate character strings generated by the abovementioned candidate character string generator, outputting an optimum candidate character string as a recognition result of the abovementioned input language element string.

[0052] According to the abovementioned configuration, an optimum candidate character string can be produced as a recognition result from a plurality of candidate character strings selected by the abovementioned probability system language processing by implementation of a word inquiry system language processing by means of a word dictionary search. An input document and input speech can be more precisely recognized in this way.

[0053] In addition, in the invention pertaining to claim 15 of the recognition apparatus pertaining to claim 13, the abovementioned score calculator calculates the score in accordance with the abovementioned 2nd equation indicated above.

[0054] According to the abovementioned configuration, the abovementioned score is calculated by a very simple calculation processing that involves reference to the abovementioned 2-cluster code transition probability table, the 3-character attribute transition probability table and the 1-character appearance probability table.

An increase in the recognition processing actuation speed can be achieved in this way.

[0055] In addition, in the invention pertaining to claim 16 of the recognition apparatus of the invention pertaining to claim 12, the abovementioned input language element string is a character string.

[0056] According to the abovementioned configuration, by using the abovementioned 2-cluster code transition probability table in which the memory capacity is compressed by an amount (cluster number/recognition object character number)^2 more than a conventional 2-character transition probability table, a high-recognition rate character recognition processing can be implemented at high speed using a small memory capacity.

[0057] In addition, on the computer-readable recording medium of the invention pertaining to claim 17, a character recognition program is recorded for making a computer function as a segmenter that segments individual characters from an input character string, a matcher that produces a plurality of recognition candidates based on matching feature patterns of segmented characters with a standard pattern, and a candidate character string generator that, by implementing a probability system language processing in which the abovementioned 2-cluster code transition probability table generated by the probability table generating apparatus described in claim 1 is employed on the plurality of candidate character strings produced by combining the abovementioned plurality of recognition candidates, calculates a score of the abovementioned candidate character strings and, in accordance with this score, generates a predetermined number of candidate strings.

[0058] According to the abovementioned configuration, similarly to the invention pertaining to claim 16, by using the abovementioned 2-cluster code transition probability table, a high-recognition rate character recognition processing can be implemented at high speed

using a small memory capacity.
[0059] Notably, the abovementioned language element refers to a concept that expresses characters or syllables serving as single elements of natural language.
[0060]
[Embodiments of the Invention] The present invention will be hereinafter described in detail with reference to the embodiments shown in the drawings. The probability table generating apparatus of this embodiment generates a compressed digram obtained by compression of character number or syllable number without loss of the linguistic transition relationship (the digram is not literally compressed but instead is regenerated as a compressed character (or compressed syllable) code digram when the recognition object character (or recognition object syllable) number is compressed: equivalent to the abovementioned cluster code transition probability table); a unigram which constitutes a table that expresses 1-character (or 1 syllable) appearance probability; and an attribute trigram which constitutes a table that expresses 3-character (or 3-syllable) attribute combination appearance probability. The small memory capacity and high-speed characteristics of the probability system language processing are realized by the employment of the abovementioned compressed digram. On the other hand, the information lost by compression of the recognition object character (or recognition object syllable) number is compensated by the combined use of the abovementioned unigram and attribute trigram and, as a result, a high-accuracy probability system language processing in which there is no loss of the small memory capacity and high-speed characteristics can be realized.
[0061] The abovementioned compressed digram and method of generation thereof, attribute trigram and method of generation thereof, as well as the method for calculating character string probability that form the

basis of the invention will be hereinafter described in detail.

[0062] (a) Regarding the compressed digram (2-cluster code transition probability table)

The memory capacity of the abovementioned digram increases proportionately to the square of the recognition object character number. Conversely, if the recognition object character number is small, the memory capacity is dramatically reduced. For example, assuming the need for 1 byte per 1 element, the memory capacity of a digram of a 1000 character recognition object is 1000 x 1000 = 1 megabyte, and a memory capacity of this magnitude is fully realizable. Accordingly, the digram itself is not compressed but rather the recognition object character number need only be compressed to around 1000 by some method. The compressed digram of this invention, which is considered with this in mind, is generated as described below.

[0063] First, the 1-character appearance probability (unigram) and, as shown in FIG. 2(a), the transition probability (digram) between adjacent 2-characters is obtained from a large-sized general document (should the recognition object text be limited, from the recognition object text thereof). Taking the recognition object character category number as 4000, at this point the capacity of the digram is 4000 x 4000 = 16 megabytes. Notably, the capacity of the unigram is 4000 = 4 kilobytes.

[0064] Next, hypothesizing the columns (transition probability from the character being considered to all characters) of the abovementioned digram as 4000-dimension vector data, the digram is decomposed into 4000 vectors. As shown in FIG. 2(b), these 4000 element vectors are clustered into a predetermined number (for example 1000). The conventionally known k-means or word techniques or similar are used to perform this clustering. Notably, the abovementioned clustering results in plurality of vectors in close proximity

being represented in the one cluster.

[0065] The abovementioned vectors use as an element for any character selected as a focus character the transition probability from a focus character to all characters. Accordingly, the linguistic transition relationship of vectors (focus characters) in which there is little distance therebetween (in other words, which have been classified in the same cluster by clustering) is very low. In other words, even if a plurality of focus characters (vectors) c1 to c4 classified in the same cluster is replaced by a single compression character code (cluster code) cc0, little linguistic information loss occurs. The 4000 recognition object characters can be compressed to 1000 compressed character codes with minimized loss of linguistic information in this way.

[0066] Subsequent to completion of the abovementioned clustering, the thus obtained compressed character codes are employed to generate a digram (compressed digram) obtained by compression of the character number again from the large-size document. Notably, in this case, the characters from the large-size document must be converted to compression character codes (cluster codes). Thereupon, a correlation table (compression character code conversion table) expressing correlation between the compression character codes and the character codes is generated from the results of the abovementioned clustering and, employing this compression character code conversion table, input characters are converted to compressed character codes. The compressed digram produced in this way has a memory capacity proportionate to the square of the compression character code number (cluster number: 1000) which, when the recognition object character number is 4000, is 1 megabyte. That is to say, a memory capacity that is fully realizable can be achieved. In addition, because the element values here are multiple values, the transition probability between the compression character codes can be represented with high accuracy.

[0067] (b) Regarding the attribute trigram (3-character attribute transition probability table)

The abovementioned attribute trigram compensates the information that is lost when the abovementioned compressed digram (more accurately, the digram in which the recognition object character number is compressed) is generated. As shown in FIG. 3, the 7 character attributes of “Attribute Hi: Hiragana”, “Attribute Ka: Katakana”, “Attribute Sym: Symbol”, “Attribute Kan: Kanji”, “Attribute Num: Numeral”, “Attribute Upp: Upper case alphabet” and “Attribute Low: Lower case alphabet” are employed in this embodiment. Accordingly, the memory capacity of the attribute trigram generated employing the abovementioned attributes is equivalent to the cube of 7 bytes, which is 343 bytes. This attribute trigram, similarly to the compressed digram, employs a character attribute conversion table to convert the character codes of characters from the large-size document into character attribute codes. The results are obtained by calculation of the character attribute code transition probability between adjacent 3-characters.

[0068] The total memory capacity of the unigram, compressed digram and attribute trigram produced in the manner described is of the order of 1 megabyte and is fully practically applicable, and these are high-accuracy probability tables in which there is little loss of language information. Accordingly, by employing these probability tables in language processing, a high-accuracy, high-speed and small memory capacity recognition processing can be implemented.

[0069] (c) Regarding the method for calculating character string transition probability

In the probability system language processing of this invention a character string transition probability P(x) of a character string x is calculated by the following approximate Equation (3) obtained from Equation (1) and Equation (2) employing the abovementioned unigram, compressed digram and attribute

trigram probability tables. Notably, the character string probability transition employed in this invention is premised on the logarithmic compression of Equation (1). Accordingly, the term “log{P(x)}” is indicated hereinafter only to establish a distinction for the purpose of the description, and a logarithmically compressed transition probability is also expressed when this term is written simply as P(x).

[0070]

$$P(c_1, c_2, \ldots, c_n) = P(c_1) + P(c_1, c_2) + P(c_1, c_2, c_3) + P(c_2, c_3, c_4) + \ldots + P(c_{n-2}, c_{n-1}, c_n) + P(c_{n-1}, c_n) + P(c_n) \quad (3)$$

[0071] Here, each of the transition probabilities is calculated by the following approximate Equation (4) employing the abovementioned unigram, compressed digram and attribute trigram.

$$P(x) = \text{unigram } (x)$$

$$P(x, y) = \{\text{unigram } (x) + \text{unigram } (y)\}/2 + \text{compressed digram } (x^c, y^c)/2$$

$$P(x, y, z) = \{\text{unigram } (x) + (\text{unigram } (y) + \text{unigram } (z)\}/3 + \{\text{compressed digram } (x^c, y^c) + \text{compressed digram } (y^c, z^c)\}/2 + \{\text{attribute trigram } (x^a, y^a, z^a)\}/3 \quad (4)$$

wherein,

x, y, z : character codes

x^c, y^c, x^c : compression character codes of character codes x, y, z

x^a, y^a, z^a : character attribute codes of character codes x, y, z

Unigram (x) : element values of character code x of unigram

Compressed digram (x^c, y^c): element values of compression character code sets (x^c, y^c) of compressed digram

Attribute trigram (x^a, y^a, z^a): element values of character attribute code sets (x^a, y^a, z^a) of attribute trigram

[0072] In a specific description thereof, a character string “本日はXXXXへ” is expanded to:

P(本日はXXXXへ) = P(本) + P(本日) + P(本日は) + P(日はシ) + P(はシャ) + …+ P(ーブへ) + P(ブへ) + P(へ), the portion P(本日は) being established as follows.

P(本日は) = {unigram (本) + unigram (日) + unigram (は)}/3 + {compressed digram (本^{c}, 日^{c}) + compressed digram (日^{c}, は^{c})}/2 + {attribute trigram (本^{a}, 日^{a}, は^{a})}/3

[0073] In the character string transition probability expansion equation described above, the division by 3 is performed in a division table and the division by 2 is realizable by means of bit shift. Accordingly, the calculation Equation described above can be performed by reference to the table and addition alone, and this in turn facilitates high-speed processing.

[0074] FIG. 1 is a function block diagram of a first embodiment of a probability table generating apparatus. This probability table generating apparatus generates the abovementioned unigram, compressed digram and attribute trigram probability tables and so on employed for executing the probability system language processing, and stores them in a memory. While this probability table generating apparatus is applicable to all language types, for convenience of the description the language used herein is Japanese, and the processing object character number is taken as 4000 characters. In addition, all Kanji codes are internally coded (1 to 4000).

[0075] A large-size general Japanese language document is stored in a document database 1. A character, the character directly preceding this character, and a character preceding this preceding character are read from the document database 1 and stored in a current character buffer 15, a previous character buffer 16 and a pre-previous character buffer 17. A unigram generator 19 calculates the appearance probability of each character of all characters of the document database 1 and stores this is in a unigram buffer 2. A digram

generator 20 calculates the appearance probability (transition probability) of the 2-character combinations of all characters in the document database 1 and stores this in a digram buffer 4. Furthermore, an attribute trigram generator 21 calculates the appearance probability (transition probability) of the 3-character attribute combinations of all characters in the document database 1 and stores this in an attribute trigram buffer 6.

[0076] Subsequent to the character number being compressed as a result of the recognition object character number being clustered into a k number of clusters as will be described in detail later, a compressed digram generator 22 calculates the appearance probability of 2-compressed character code combinations of compressed character codes (cluster codes) to which all characters of the document database 1 belong and stores this in a compressed digram buffer 8. A vector divider 23 hypothesizes the columns of the abovementioned digram as vector data, divides this into vectors, and stores the divided vectors in a vector buffer 10. A clusterer 24 clusters the abovementioned vectors of all characters and stores the cluster results thereof in a cluster buffer 11. The abovementioned probability tables are generated as a result of a control of the unigram generator 19, digram generator 20, attribute trigram generator 21, compressed digram generator 22, vector divider 23 and clusterer 24 and so on executed by a controller 18.

[0077] Notably, a total character number counter 3 counts the total number of characters in the abovementioned document database 1. A total 2-character combination number counter 5 counts the total number of 2-character sets present in the document database 1. A total 3-character attribute combination number counter 7 counts the total number of 3-character attribute sets present in the document database 1. A total 2-compressed character code combination number counter 9 counts the total number of 2-compressed character code

sets present in the document database 1. In addition, conversion of the characters of the document database 1 into character attributes involves reference to a character attribute conversion table 12. Conversion of characters in the document database 1 into compressed character codes (in other words, compression of the character number of the document database 1) involves reference to a compressed character code conversion table 13. A temporary memory 14 is used to temporarily store the data produced by the processings described above.

[0078] The probability table generating apparatus of the configuration described above is actuated to generate the probability tables as a result of a control executed by the abovementioned controller 18 in the following way. FIG. 4 and FIG. 5 are flow charts of the actuation of the probability table generation processing executed by the abovementioned controller 18. The steps for generating the probability tables will be hereinafter described in detail with reference to FIGS. 4 and 5.

[0079] In Step S1, all the buffers and all the counters are cleared to "0". In Step S2, 1-character is read from the document database 1 and stored in the current character buffer 15. In Step S3, the character is judged as being present or not in the abovementioned document database 1. If present, then processing shifts to Step S4, and if not, then processing shifts to Step S14.

[0080] In Step S4, the total character number is counted by the abovementioned total character number counter 3. In Step S5, a unigram computation is performed by the unigram generator 19. This unigram computation is based on incrementing the element values (appearance number) of the unigram buffer 2 correspondent to the current characters stored in the current character buffer 15.

[0081] In Step S6, the character code of the previous character is judged as stored or not in the

abovementioned previous character buffer 16. If stored, then processing shifts to Step S7, and if not, then processing shifts to Step S9. In Step S7, the total number of 2-character sets is counted by the total 2-character combination number counter 5. In Step S8, a digram computation is performed by the digram generator 20. This digram computation is based on incrementing the element values (appearance number) of the digram buffer 4 correspondent to the 2-character combinations stored in the previous character buffer 16 and the current character buffer 15.

[0082] In Step S9, the character code of the character previous to the previous character is judged as stored or not in the abovementioned pre-previous character buffer 17. If stored, then processing shifts to Step S10, and if not, then processing shifts to Step S13. In Step S10, the character codes stored in the current character buffer 15, previous character buffer 16 and pre-previous character buffer 17 are converted to character attribute codes. This character attribute conversion involves direct conversion from the character codes (the kanji codes are all internally coded from 1 to 4000) employing the abovementioned character attribute conversion table 12. The character attribute conversion table 12, which has a structure that uses the character code as an offset address to facilitate reference to the character attribute code of a character code in question, requires a (recognition object character number x 1 = 4000) byte capacity. Notably, the character attribute conversion table 12 is generated in advance.

[0083] In Step S11, the total number of 3-character attribute sets is counted by the abovementioned total 3-character combination number counter 7. In Step S12, an attribute trigram computation is performed by an attribute trigram generator 21. This attribute trigram computation is based on incrementing the element values (appearance number) of the attribute trigram buffer 6 correspondent to the 3-character character attribute

combinations stored in the pre-previous character buffer 17, previous character buffer 16 and current character buffer 15.

[0084] In Step S13, the contents of the pre-previous character buffer 17 and previous character buffer 16 are updated by the contents of the abovementioned previous character buffer 16 and the current character buffer 15. Thereafter, the processing returns to the abovementioned Step S2 for processing of the next 1-character. If the character has already been judged in the abovementioned Step S3 as not being present in the document database 1, in Step S14, the element values (appearance numbers) of the abovementioned unigram, digram and attribute trigram are converted to probability values by the unigram generator 19, digram generator 20 and attribute trigram generator 21. The conversion to probability values involves division of the element values of the abovementioned unigram, digram and attribute trigram by the counted values of the total character number counter 3, the total 2-character combination number counter 5, and the total 3-character combination number counter 7.

[0085] However, the element values (probability values) of the unigram, digram and attribute trigram calculated in this way are taken to the decimal place and have a large number of significant digits. Accordingly, the probability system language processing tabulation performed in this state leads to an increase in the memory capacity and, because the calculation of the character string transition probability involves multiplication, the calculation speed also drops. Thereupon, in Step S15, a logarithmic compression is performed on the element values of the abovementioned unigram buffer 2, digram buffer 4 and attribute trigram buffer 6 converted to probability values in the abovementioned Step S14, and the element values in question are updated by the thus produced logarithmic compression values.

[0086] The abovementioned logarithmic compression is

performed in accordance with the following equation.

$$\ln P(x) = \min(255, (-\log(P(x)) - MIN)) \times 256/(MAX - MIN) \quad (5)$$

Here, P(x) denotes the probability value of an arbitrary character x, the value P(x)>0 being adopted. In addition, if P(x)=0, lnP(x)=255. Notably, min(a, b) constitutes functions for which the minimum values of a and b are adopted. In addition, MIN and MAX denote the minimum and maximum probability values subsequent to the abovementioned logarithmic compression and in this embodiment these values are “0” and “18” respectively.

[0087] Based on the abovementioned Equation (5), the probability value P(x) in the range “1 to 0.000000000000000001” is converted for 0 to 255 to 1-byte data. Notably, low probability values of -log(P)>MAX(=18) or lower (also includes infinity -log(0)) are suppressed to the maximum value representable using 1 byte of “255”.

[0088] While a larger probability value normally suggests greater likelihood, the probability value in this embodiment is given as an estimated value for which, because of the logarithmic conversion performed, a lower value expresses greater likelihood.

[0089] Subsequent to the abovementioned digram being obtained in this way, the processing shifts to the calculation of the abovementioned compressed digram.

[0090] In Step S16, the following vector division processing is executed by the abovementioned vector divider 23. That is to say, hypothesizing the digram stored in the digram buffer 4 as a 4000 x 4000 2-dimensional array (4000 x 4000 matrix), the columns are divided as 4000-dimension vectors into a 4000 vectors and stored in the vector buffer 10. These vectors constitutes 4000-dimension vectors for which transition probability from a focus character to all characters serves as the element thereof.

[0091] In Step S17, the abovementioned clusterer 24 clusters the abovementioned divided 4000 vectors employing a k-means or word clustering technique. The

maximum cluster number in this case in this embodiment is set as “1000”. The thus produced clusters are stored in the cluster buffer 11.

[0092] In Step S18, the compressed character code conversion table 13 used for converting the abovementioned character code to the compressed character code is generated. The compressed character code conversion table 13 is generated in the following way. That is to say, generation of the compressed character code conversion table 13 is based on generation of a table in which there is correlation between character codes c1 to c4, c5 to c8, …, c3997 to c4000 of each of the focus characters of the vectors accumulated in each cluster and the cluster codes of these clusters (in other words, the compression code cc0 to cc999). Similarly to the character attribute conversion table 12, the compressed character code conversion table 13 constitutes a structure in which, in order to ensure direct conversion from the character code to the compressed character code, the character codes (1 to 4000) are used as an offset address to facilitate reference to the compression character code of a character code in question. The memory capacity of the compressed character code conversion table 13 is 4000 x 2 = 8 kilobytes.

[0093] In Step S19, the abovementioned compressed digram buffer 8, the total 2-compressed character code combination number counter 9, the current character buffer 15 and the previous character buffer 16 is cleared to “0”. In Step S20, the 1-character read from the document database 1 is stored in the current character buffer 15. In Step S21, the character is judged as present or not in the abovementioned document database 1. If present, then the processing shifts to Step S22, and if not, then the processing shifts to S27.

[0094] In Step S22, the previous character is judged as stored or not in the abovementioned previous character buffer 16. If stored, then the processing shifts to

Step S23, and if not, then the processing shifts to Step S26. In Step S23, the character codes stored in the current character buffer 15 and the previous character buffer 16 are converted to compressed character codes. This compressed character code conversion involves direct conversion from the character codes to the compression character codes employing the compressed character code conversion table 13.

[0095] In Step S24, the total number of 2-compresed character code sets is counted by the abovementioned total 2-compressed character code combination number counter 9. In Step S25, a compressed digram computation is performed by the compressed digram generator 22. This compressed digram computation is based on incrementing the element values (appearance number) of the compressed digram buffer 8 correspondent to the 2-compressed character code combinations produced in Step S23 described above.

[0096] In Step S26, the contents of the previous character buffer 16 are updated by the contents of the abovementioned current character buffer 15. Thereafter, the processing shifts to the abovementioned Step S20 for processing of the next 1-character. In addition, when a judgment that the character is not present in the document database 1 has already been made in Step S21 described above, in Step S27, the element values (appearance number) of the abovementioned compressed digram are converted to probability values by the compressed digram generator 22. This conversion to probability values involves the element values of the abovementioned compressed digram being divided by the counted value of the total 2-compressed character code combination number counter 9. In addition, similarly to the case of the abovementioned unigram, digram and attribute trigram, in Step S28, a logarithmic compression is performed on the element values of the compressed digram buffer 8 converted to probability values in Step S27 described above, and the element

values in question of the abovementioned compressed digram buffer 8 are updated by the thus produced logarithmic compression values. Subsequent to the abovementioned compressed digram being generated in this way, the actuation of the probability table generation processing ends.

[0097] The memory size of the probability tables generated in this way are shown in FIG. 6, the sum total of the memory capacity of the abovementioned unigram, compressed digram, attribute trigram, character attribute conversion table 12 and compressed character code conversion table 13 being highly compacted to approximately 1 megabyte.

[0098] In this embodiment the total number of character categories in the document database 1 is counted by the total character number counter 3, the total number of 2-character sets is counted by the total 2-character combination number counter 5, and the total number of 3-character attribute sets is counted by the total 3-character combination number counter 7.

[0099] The abovementioned unigram generator 19 updates the element values for which there is correlation with the unigram buffer 2 based on calculation of the appearance number of all characters and, when there are no characters remaining in the document database 1, converts the element values (appearance number) of the unigram buffer 2 to probability values employing the counted values of the abovementioned total character number counter 3. In addition, the abovementioned digram generator 20 updates the element values for which there is correlation with the unigram buffer 4 based on calculation of the appearance number of all 2-character sets to, when there are no characters in the document database 1, converts the element values (appearance number) of the digram buffer 4 to probability values employing the counted values of the abovementioned total 2-character combination number counter 5. The attribute trigram generator 21, in accordance with the character attribute codes obtained

by conversion of the pre-previous character, pre-character and current character character codes, updates the element values for which there is correlation with the attribute trigram buffer 6 based on calculation of the appearance number of all 3-character attribute sets and, when there are no characters in the document database 1, converts the element values (appearance number) of the attribute trigram buffer 6 to probability values employing the counted values of the abovementioned total 3-character combination number counter 7. The element values of the unigram buffer 2, digram buffer 4 and attribute trigram buffer 6 are then logarithmically compressed to form the abovementioned unigram, digram and attribute trigram.

[0100] Subsequent to the abovementioned digram being obtained in this way, the abovementioned digram, which constitutes a 4000 x 4000 matrix, is divided into 4000 4000-dimension vectors by the vector divider 23. These vectors are clustered by the clusterer 24, and the compressed character code conversion table 13 in which there is correlation between the character codes belonging to each cluster and the cluster codes (compression character codes) is generated. Thereafter, the current character buffer 15 and the previous character buffer 16 are updated each time a 1-character is read from the document database 1. Furthermore, the current character and previous character character codes are converted to compressed character codes employing the compressed character code conversion table 13, and the total number of 2-compressed character codes is counted by the total 2-compressed character code combination number counter 9. Similarly to the digram generation described above, the compressed digram is generated by the compressed digram generator 22.

[0101] The abovementioned digram generated in this way compresses the element number by 1/16 without loss of the linguistic transition information possessed by the

abovementioned digram. Accordingly, by implementing a language processing employing the abovementioned compressed digram, small memory capacity and high-speed characteristics can be realized without loss of language information. In addition, the linguistic transition information lost by the compression of the abovementioned recognition object character number is compensated by the combined use of the unigram which expresses 1-character appearance probability and the attribute trigram which expresses the character attribute code transition probability between adjacent 3-characters which, in turn, facilitates a high-accuracy probability system language processing in which there is no loss of the abovementioned small memory capacity and high-speed characteristics.

[0102] That is to say, according to this embodiment, a probability system language processing of small memory capacity of the order of 1 megabyte and which employs high-accuracy multiple-value probability tables can be performed even on languages having a large number of recognition object characters such as the Japanese language. In addition, the character string probability calculation in this case involves a very simple calculation based on addition and referral to a table which, in turn, facilitates high-speed processing. Accordingly, a high-recognition rate recognition apparatus in which high-speed processing is possible using a small memory capacity can be realized.

[0103] FIG. 7 is a block diagram of a second embodiment of a recognition apparatus. This recognition apparatus, which comprises probability system language processing means in which the abovementioned probability table generated by the first embodiment is employed, performs recognition of characters from input document image data. Notably, while this recognition apparatus has application for all language types, for convenience of the description the language of Japanese, a recognition object character number of 4000 characters, and the use of Japanese language typeface OCR that employs

similarity as a recognition estimate value is assumed. In addition, all kanji codes are taken as being internally coded (1 to 4000).

[0104] An image inputter 31 uptakes document image data from a scanner 47 or an image file 48 and stores it in an image memory 49. A segmenter 32 comprises an area segmenter 33 for segmenting areas in which the character recognition is to be performed in accordance with a 1-document image data stored in the image memory 49, a column segmenter 34 for segmenting 1-column character strings, a character segmenter 35 for segmenting 1-characters, and a touch-character segmenter 36 for forcibly separating the touching characters. Notably, the coordinates of the thus produced abovementioned area, columns and characters on the image memory 49 are stored in an area coordinate memory 50, a column coordinate memory 51 and a (temporary) character coordinate memory 52 respectively. In addition, later-described column direction and row direction peripheral distribution features are stored in a peripheral distribution feature buffer 53. A feature extractor 37 extracts feature patterns in accordance with the abovementioned segmented 1-character image data and stores this in a feature memory 54.

[0105] A matcher 38 comprises a first similarity calculator 39 for calculating similarity based on matching of feature patterns of the abovementioned segmented 1-characters with a standard pattern of recognition object characters stored in a pattern dictionary 55, and a first sorter 40 for producing recognition result candidates based on sorting of the abovementioned calculated similarities in descending order and selection of a predetermined number of character candidates. Notably, the thus produced recognition result candidates are stored in a recognition result candidate buffer 56.

[0106] A character string generator 41 comprises a candidate character string expander 42 for combining

and expanding all recognition result candidates for a character in question to a candidate character string each time 1-character recognition result candidates from which the abovementioned 1-column are configured is produced, a score calculator 43 for calculating the score of the thus produced candidate character string employing a table 58 such as, for example, the abovementioned unigram, compressed digram and attribute trigram, and a second sorter 44 for sorting the calculated scores in descending order and selecting a predetermined number of candidate character strings.

[0107] A word inquiry language processor 45 performs a word inquiry system language processing on a 1-column segment of the abovementioned predetermined number of candidate character strings stored in the candidate character string buffer 57 employing a word inquiry language dictionary 59. A final recognition result obtained in accordance with the result of the word inquiry language processing is used to update the contents of a recognition result buffer 60. This character recognition processing is implemented by a control of the image inputter 31, segmenter 32, feature extractor 37, matcher 38, character string generator 41 and word inquiry language processor 45 executed by a controller 46.

[0108] That is to say, the abovementioned probability system language processing means of this embodiment is configured from the abovementioned score calculator 43 and second sorter 44.

[0109] The recognition apparatus of the configuration described above performs the character recognition processing as a result of actuation by a control executed by the abovementioned controller 46 in the following way. FIG. 8 is a flow chart of the actuation of the character recognition processing executed by the abovementioned controller 46. The steps for character recognition will be hereinafter described with reference to FIG. 8.

[0110] In Step S31, a 1-document image data uptaken

from the scanner 47 or image filer 48 by the abovementioned image inputter 31 is stored in the image memory 49. In Step S32, the total area in which the character recognition is to be performed is segmented from the abovementioned 1-document by the area segmenter 33. Otherwise, the abovementioned total area is designated. The coordinates of the segmented area or designated area in the image memory 49 are stored in the area coordinate memory 50. In Step S33, the image data of one area is read. In Step S34, the presence or not of the area is determined. If present, then the processing shifts to Step S35, and if not, then the processing shifts to Step S49.

[0111] In Step S35, all columns are segmented from the area in question by the abovementioned column segmenter 34. There are no particular limitations to the column segmenting method used here. For example, the pixel number of a predetermined brightness or above in the column direction of the image data of the area in question is counted and stored in the peripheral distribution feature buffer 53 as the abovementioned peripheral distribution feature, and the segmentation is performed in accordance with this peripheral distribution feature. In Step S36, the presence or not of the segmented column is determined. If present, then the processing returns to the abovementioned Step S33 for processing of a next area, and if not, then the processing shifts to Step S37.

[0112] In Step S37, a one column image data of is read. In Step S38, a temporary character segmentation is performed in accordance with the abovementioned read 1-column image data. Notably, the abovementioned “temporary character segmentation” processing does not necessarily result in segmentation of the correct answer characters and, taking the kanji character “加 ” as an example, the processing involves extraction of all locations that segment "加 ” or “力” and “口” as 1-characters. There are no particular limitations to the method for temporary character segmentation used here.

For example, the pixel number of a predetermined brightness or above in the row direction of the image data of the line in question is counted and stored in the peripheral distribution feature buffer 53 as the abovementioned peripheral distribution feature, and the segmentation is performed in accordance with this peripheral distribution feature. Furthermore, locations that can be forcibly separated are extracted by the touch-character segmenter 36 from a character clump of touching characters employing the abovementioned row direction peripheral distribution feature stored in the peripheral distribution feature buffer 53.

[0113] In Step S39, the abovementioned candidate character string buffer 57 is intialized. In Step S40, the presence or not of the character in the column in question is determined. If present, then the processing shifts to Step S41, and if not, then the processing shifts to Step S47. In Step S41, an actual segmentation of the 1-characters from the column in question is performed by the character segmenter 35 in accordance with the abovementioned temporary character segmentation result.

[0114] In Step S42, the feature patterns of the abovementioned segmented character candidates in question are extracted by the abovementioned feature extractor 37. In Step S43, the feature patterns of the character candidates in question and the standard pattern of the recognition object character stored in the pattern dictionary 55 are matched and a similarity is calculated by the first type similarity calculator 39. Furthermore, the abovementioned calculated similarities are sorted in descending order and a predetermined number (“10” in this embodiment) of character candidates selected by the first sorter 40. The thus produced character candidates are stored in the recognition result candidate buffer 56.

[0115] Thereafter, the character candidates stored in the abovementioned recognition result candidate buffer 56 are combined and expanded to form character strings

which are stored in the candidate character string buffer 57. Here, storage of all character candidate combinations in the candidate character string buffer 57 is impossible because this necessitates a 10-fold (total 1-column character number) increase in the candidate character string memory capacity. A marked drop in processing speed also results. For this reason, in this embodiment a score of all candidate character strings is obtained as follows, and the candidate character string number is reduced in accordance with this score.

[0116] In Step S44, all character candidates stored in the abovementioned recognition result candidate buffer 56 are combined and expanded by the abovementioned candidate character string expander 42 to form a character string. In Step S45, scores of all the thus produced abovementioned candidate character strings are calculated by the abovementioned score calculator 43. The calculation of the scores is performed in accordance with Equation (6).

Score (1) = $W_S S_1 + W_P P(c_1)$

Score (2) = $W_S\{(S_1 + S_2)/2\} + W_P P(c_1, c_2)$

Score (i) = [Score (i-1) x (i-1)

$+ \{W_S S_i + W_P P(c_{i-2}, c_{i-2}, c_i)\}]/i$

Here,

Score (I) : Score of candidate character strings of character candidate number I

$c_1, c_2, c_3, \ldots, c_n$: Candidate character strings

$S_1, S_2, S_3, \ldots, S_n$: Similarities of character candidates

P(x) : Character string transition probability of candidate character string x (calculation performed employing Equation (4) of the first embodiment)

W_P, W_S : Weightings

Notably, it is desirable that the abovementioned weightings W_P, W_S satisfy the relationship $W_S > W_P$.

[0117] The scores in this embodiment represent the weighted sum of the similarity S_n, which is calculated

by the second similarity calculator, and the transition probability P(x), which is calculated by the probability calculator, of the abovementioned score calculator 43. Accordingly, even when an estimate scale based on recognition estimate values other than that of the abovementioned similarity S_n and the abovementioned transition probability P(x) is used, provided the value obtained by multiplication of the abovementioned estimate scale with the weighting of the value is added as per the third clause of Equation (6), a score calculation in which the abovementioned unigram, compressed digram and attribute trigram are employed can be performed.

[0118] In Step S46, the abovementioned calculated candidate character strings are sorted in descending order and a predetermined number (“100” in this embodiment) of candidate character strings selected in order of largest score by the abovementioned second sorter 44. The thus selected 100 candidate character strings are stored in the candidate character string buffer 57. Thereafter, the processing returns to the abovementioned Step S40 for segmentation of the next 1-character. Next, until there are no characters in the column in question, the character segmentation from the column in question, the feature extraction of the segmented characters, the matching of the segmented characters, and the generation of the candidate character strings on the basis of the already produced character candidates and the recognition candidates of the segmented characters (score calculation and character reduction processing) are performed in sequence. Thereupon, subsequent to the determination in the abovementioned Step S40 that there are no characters in the column in question, the processing shifts to Step S47.

[0119] In Step S47, a word inquiry-type language processing is executed on the 1-column segment of 100 candidate character strings stored in the candidate character string buffer 57 by the abovementioned word

inquiry language processor 45 employing the word inquiry language dictionary 59. In Step S48, the optimum candidate character string is selected from the 1-column segment of 100 candidate character strings stored in the candidate character string buffer 57 in accordance with the abovementioned score and the word number present in the word inquiry language dictionary 59, and this is stored in the recognition result buffer 60.

[0120] Thereafter, the processing returns to the abovementioned Step S36 where the following column processings are performed. Subsequently, and until there are no columns in the area in question, the character segmentation from the columns in sequence, the feature extraction of the segmented characters, matching of the segmented characters, the generation of the candidate character strings in accordance with the already produced character candidates and the recognition candidates of the segmented characters (score calculation and candidate reduction processing), word inquiry-system language processing, and the selection of the optimum candidate character string are performed in sequence. In addition, subsequent to the judgment in the abovementioned Step S36 that there are no columns in the area in question and the determination in the abovementioned Step S34 that there are no areas, the processing moves to Step S49. In Step S49, the optimum candidate character string of all columns stored in the recognition result buffer 60 is output as a recognition result. Thereafter, the actuation of the character recognition processing ends.

[0121] The probability tables (unigram, compressed digram and attribute trigram) thus generated in this embodiment by the probability table generating apparatus of the first embodiment are loaded in a recognition apparatus (Japanese typeface OCR). Then, 1-characters are segmented by the segmenter 32 from the image data of a document uptaken by the image inputter 31, feature patterns are extracted by the feature

extractor 37, and 10 character candidates are produced by the matcher 38. Subsequent to 10 each of the character candidates for all 1-column characters being produced in this way, all the character candidates are combined by the candidate character string expander 42 and expanded to form candidate character strings.

[0122] Subsequently, a score of all the abovementioned candidate character strings produced by the abovementioned score calculator 43 is calculated employing the abovementioned probability table in accordance with Equation (6). In addition, all the candidate character strings are sorted in descending order of score and the 100 candidate character strings selected in order of largest score by the second sorter 44.

[0123] By implementing the abovementioned probability system language processing in this way, the 100 candidate character strings of most likelihood, without recourse to word segmentation (morpheme analysis), are able to be quickly selected from a large number of candidate character strings generated as a result of expanding of the character candidates for which 10 each are produced for all of the 1-column characters. The abovementioned probability tables employed here are configured from the abovementioned unigram, compressed digram and attribute trigram of the first embodiment. Accordingly, the probability system language processing implemented by this embodiment is executed with high accuracy and at high speed using small memory capacity.

[0124] Next, the word inquiry-system language processing is implemented on the abovementioned selected 1-column segment 100 candidate character string by the abovementioned word inquiry language processor 45 in which the word inquiry language dictionary word inquiry language dictionary 59 is employed, and the optimum candidate character string is selected as a recognition result.

[0125] That is to say, according to this embodiment, a probability system language processing in which a high

accuracy multiple-value probability tables of small memory capacity of the order of 1-megbyte is employed can have application in the recognition of languages in which the recognition object character number is large such as Japanese, and a high-recognition rate recognition apparatus in which high speed processing using a small memory capacity is possible can be realized.

[0126] Notably, while the first embodiment cites the example of application to Japanese language characters, application to all languages is possible. This language processing is particularly effective for languages of large character number such as Japanese and Chinese. Moreover, examples of the abovementioned character attributes that should be employed for the generation of a Chinese language recognition probability table include joji, symbols, kanji other than joji, numerals, alphabet upper case and alphabet lower case letters.

[0127] Furthermore, because the character string probability calculation in which the abovementioned probability table generated by the first embodiment is employed is effective for determining the likelihood of a kanji/kana mixed character string, application in a language processing apparatus such as a kanji/kana conversion apparatus is possible. In addition, while in the first embodiment a probability table employed for character recognition or kana-kanji conversion is generated in accordance with a Japanese language document stored in the document database 1, a probability table employed for speech recognition can also be generated. In this case, the Japanese language kana character string is stored in the document database 1, and the abovementioned compressed digram, unigram and attribute trigram are produced employing this kana character string.

[0128] In addition, the probability table of the second embodiment may be generated by a probability table generating apparatus obtained by loading of probability table generating apparatus of the first embodiment in a

recognition apparatus. Otherwise, a probability table generated by another probability table generating apparatus may be installed in this recognition apparatus. In addition, it is clear that the second embodiment is not restricted to Japanese language characters and can have application in recognition apparatuses for other language characters. Furthermore, it is not restricted to character recognition and can have application in speech recognition.

[0129] In addition, while generation or loading of all of the abovementioned compressed digram, unigram and attribute trigram is possible in both the first and second embodiments, in this invention generation or loading of at least the abovementioned compressed digram is sufficient.

[0130] In addition, the programs for the abovementioned probability table generating processing of the probability table generating apparatus of the first embodiment may be recorded in a ROM (Read Only Memory) (not shown in the digram) or RAM (Random Access Memory) (not shown in the digram) by any of the following methods.

(a) Recording in the abovementioned ROM in advance.

(b) Storing of all or some of the abovementioned probability table generating apparatus programs in a recording medium such as a floppy disk or hard disk, and installing the abovementioned programs in the abovementioned RAM in accordance with need.

(c) Installing of the abovementioned probability table generating apparatus programs in the abovementioned RAM by way of a computer network.

[0131] Similarly, the programs for the abovementioned character recognition processing of the recognition apparatus of the second embodiment may be recorded in a ROM (Read Only Memory) (not shown in the digram) or RAM (Random Access Memory) (not shown in the digram) by any of the following methods.

(a) Recording in the abovementioned ROM in advance.

(b) Storing of all or some of the abovementioned

character recognition processing programs in a recording medium such as a floppy disk or hard disk, and installing the abovementioned programs in the abovementioned RAM in accordance with need.
(c) Installing of the abovementioned character recognition processing programs in the abovementioned RAM by way of a computer network.
[0132]
[Effect of the Invention] As is clear from the above description, in the probability table generating apparatus of the invention pertaining to claim 1, because a clusterer clusters all characters of a character string stored in a memory into clusters using transition characteristics as a criteria and assigns cluster codes to these clusters, the number of recognition object characters and syllables is compressed to the abovementioned number of clusters. In addition, because a 2-cluster code transition probability table (hereinafter cluster code digram) generates a cluster digram that expresses the transition probability between all adjacent 2-character cluster codes of the characters strings sorted in the abovementioned memory, compared to a conventional digram that expresses the transition probability between adjacent 2-characters, the element number can be compressed to (cluster number/recognition object character (or syllable) number)^2. Accordingly, a small memory capacity probability table can be generated, and a small memory capacity, high-speed probability system language processing is facilitated.
[0133] In this case, because the characters or syllables belonging to the same cluster have the same transition characteristics, the linguistic transition information of the characters and syllables is retained in the abovementioned cluster digram. Accordingly, a high-speed multiple value cluster code digram in which the drop in accuracy that accompanies the compression of the total number of characters and syllables of a recognition object is suppressed can be generated.

Accordingly, a high-accuracy probability system language processing is facilitated.

[0134] In addition, in the probability table generating apparatus of the invention pertaining to claim 2, because a 3-character attribute transition probability table (hereinafter referred to as an attribute trigram) generator generates an attribute trigram that expresses the transition probability between all adjacent 3-character attributes of the character strings stored in the memory, a probability table that expresses the transition information between adjacent 3-characters and compensates the character or syllable linguistic transition information that is lost when the total number of characters and syllables of a recognition object is compressed can be produced. Accordingly, by the combined use of the abovementioned attribute trigram in the abovementioned cluster digram, the accuracy of the probability system language processing can be further improved.

[0135] In addition, in the probability table generating apparatus of the invention pertaining to claim 3, because a 1-character appearance transition probability table (hereinafter referred to as unigram) generates a unigram that expresses an appearance probability of the characters in the character string stored in the abovementioned memory, a probability table that expresses the transition information between characters can be produced to compensate for the character or syllable linguistic transition information lost when the total number of character and syllables of a recognition object are compressed. Accordingly, by the combined use of the abovementioned unigram and attribute trigram in the abovementioned cluster digram, the accuracy of the probability system language processing can be further improved.

[0136] In addition, in the probability table generating apparatus of the invention pertaining to claim 4, because the element values of the abovementioned cluster code digram are logarithmically compressed by a

logarithm compressor, the data length of the abovementioned element values can be compressed to a length that is representable using 1 byte. Accordingly, a further reduction in memory capacity of the abovementioned cluster digram is achieved and, accompanying this, a further increase in speed of the abovementioned probability system language processing is achieved.

[0137] In addition, in the probability table generating apparatus of the invention pertaining to claim 5, because a 2-character transition probability table generator generates a digram in accordance with characters strings sorted in the abovementioned memory and, because the abovementioned digram is divided into recognition object character number vectors of a recognition object character number dimension and all thus produced vectors are clustered into a predetermined number of clusters by the abovementioned clusterer, characters or syllables of similar transition characteristics can be clustered in the same cluster. Accordingly, by assigning cluster codes to the abovementioned clusters and reckoning the cluster codes as characters, the total character or syllable number of a recognition object can be compressed to the abovementioned cluster number without loss of the linguistic transition information of the characters and syllables.

[0138] Furthermore, because the cluster code conversion table generator generates a cluster code conversion table in accordance with the result of the abovementioned clustering, the abovementioned character strings are converted to the abovementioned cluster codes employing the abovementioned cluster code conversion table by the cluster code converter, and the abovementioned cluster code digram generator generates the abovementioned cluster code digram employing the converted cluster codes, the cluster code digram can be generated by production of the abovementioned cluster codes based on a simple processing involving reference

to the abovementioned cluster code conversion table.
[0139] In addition, in the probability table generating apparatus of the invention pertaining to claim 6, because an attribute converter converts the abovementioned character strings to attributes employing an attribute conversion table and the abovementioned attribute trigram generator generates the abovementioned attribute trigram employing the abovementioned converted attributes, the attribute trigram can be generated by production of the abovementioned attributes based on a simple processing involving reference to the abovementioned attribute conversion table.
[0140] In addition, in the probability table generating apparatus of the invention pertaining to claim 7, because a Japanese language document is stored in the memory, the characters are Japanese language characters, and the attributes are any of hiragana, katakana, symbols, kanji, numerals, alphabet upper case and alphabet lower case characters, the memory capacity of a Japanese language character digram for which, taking 1 element as 1 byte, 16 megabytes is required, can be compressed to 1 megabyte by generating the abovementioned cluster code digram based on character number compression to the abovementioned cluster number of 1000. Similarly, the storage capacity of the 3-character transition probability table for which 64 megabytes is required can be compressed to a maximum 343 bytes by generating the abovementioned attribute trigram based on conversion of the recognition object characters to the abovementioned attributes. That is to say, according to this invention, a small memory capacity probability table that expresses the transition information between adjacent 2-characters and adjacent 3-characters and that is loadable in a Japanese language character recognition apparatus can be generated.
[0141] In addition, in the probability table generating apparatus of the invention pertaining to claim 8,

because a Chinese language document is stored in the memory, the characters are Chinese language characters, and the attributes are any of joji, symbols, kanji other than joji, numerals, alphabet upper case and alphabet lower case characters, similarly to Japanese language, in the case of Chinese language in which the number of recognition object characters is very large, the memory capacity of the digram can be compressed to, for example, 1 megabyte, by generation of the abovementioned cluster digram by character number compression based on clustering of the recognition object characters. Similarly, the storage capacity of the 3-character attribute transition probability table can be compressed to, for example, a maximum of 216 bytes by generation of the abovementioned attribute trigram based on conversion of the recognition object characters to any of the abovementioned attributes. According to this invention, a small memory capacity probability table that expresses the transition information between adjacent 2-characters and adjacent 3-characters and that is loadable in a Chinese language character recognition apparatus can be generated.

[0142] In addition, because the probability system language processing apparatus of the invention pertaining to claim 9 comprises the abovementioned cluster code digram generated by the abovementioned probability table generating apparatus of the invention pertaining to claim 1, and because a character string transition probability calculator calculates the character string transition probability of input character strings employing the abovementioned cluster code digram compressed by an amount (cluster number/recognition object character (or syllable) number)^2 more than a conventional digram, even if the processing object constitutes Japanese language characters of recognition object character number of the order of 4000, taking 1 element as 1 byte, a 1-megabyte memory capacity is sufficient.

[0143] In addition, because the probability system

language processing apparatus of the invention pertaining to claim 10 comprises at least one of the abovementioned unigram and the abovementioned attribute trigram generated by the probability table generating apparatus pertaining to claim 2 and claim 3, and because the abovementioned character string transition probability calculator also employs the abovementioned probability tables to calculate the character string transition probability of the input character string, high-accuracy character string transition probabilities can be calculated by the combined use of at least one of the abovementioned unigram and the abovementioned attribute trigram.

[0144] In addition, because the abovementioned character string transition probability calculator of the probability system language processing apparatus of the invention pertaining to claim 11 calculates the abovementioned character string transition probability in accordance with the abovementioned 1st equation, the abovementioned character string transition probability can be calculated by a very simple calculation processing involving reference to the abovementioned cluster code digram, attribute digram and unigram. A further increase in the actuation speed of the probability system language processing is achieved in this way.

[0145] In addition, in the recognition apparatus of the invention pertaining to claim 12, because a plurality of candidate character strings are produced by combining a plurality of recognition candidates produced as a result of the matching of language elements segmented from an input language element string, a score of the abovementioned candidate character strings is calculated by a score calculator employing the abovementioned cluster code digram generated by the probability table generating apparatus of the invention pertaining to claim 1, and a predetermined number of candidate character strings are generated by a candidate character string generator

based on implementation of a probability system language processing for estimating the likelihood of the abovementioned candidate character strings in accordance with the abovementioned calculated score, a probability system language processing can be implemented on the abovementioned candidate character strings using the abovementioned cluster code digram in which a memory capacity of (cluster number/recognition object character (or syllable) number)^2 more than a conventional digram is compressed. Accordingly, even when Japanese language characters of recognition object character number of the order of 4000 are to be recognized, taking 1 element as 1 byte, a memory capacity of 1 megabyte is sufficient, and there are no impediments at all to the loading of a word inquiry-system language processing dictionary.

[0146] In addition, because the recognition apparatus of the invention pertaining to claim 13 comprises at least one of the abovementioned unigram and attribute trigram generated by the probability table generating apparatus of the invention claimed in claim 2 and claim 3, and because the abovementioned score calculator also employs the abovementioned probability tables to calculate the abovementioned candidate character string score, a score that more accurately expresses the likelihood of the abovementioned candidate character string can be calculated by the combined use of at least one of the abovementioned unigram and attribute trigram.

[0147] In the recognition apparatus of the invention pertaining to claim 14, because an optimum candidate character string is output as a recognition result of the abovementioned input language element string based on a word inquiry system language processing implemented on a predetermined number of candidate character strings generated by the abovementioned candidate character string generator by a word inquiry language processing apparatus, the optimum candidate character string can be produced as a recognition

result by means of a word dictionary search. Accordingly, both input documents and input speech can be more precisely recognized.

[0148] In addition, because the abovementioned score calculator of the recognition apparatus of the invention pertaining to claim 15 calculates the abovementioned score in accordance with the abovementioned 2nd equation, the abovementioned score can be calculated by a very simple calculation processing that involves reference to the abovementioned cluster code digram, attribute trigram and unigram. An increase in the recognition processing actuation speed can be achieved in this way.

[0149] In addition, because the abovementioned input language element string of the recognition apparatus of the invention pertaining to claim 16 is a character string, a high-recognition rate character recognition processing can be implemented at high speed using a small memory capacity based on the use of the abovementioned cluster code digram in which the memory capacity is compressed by an amount (cluster number/recognition object character number)^2 more than a conventional 2-character transition probability table.

[0150] In addition, because the computer-readable recording medium of the invention pertaining to claim 17 is one in which a character recognition program for making a computer function is recorded as a segmenter that segments individual characters from an input character string, a matcher that produces a plurality of recognition candidates based on matching feature patterns of segmented characters with a standard pattern, and a candidate character string generator that generates a predetermined number of candidate strings in accordance with the abovementioned candidate character string scores calculated for the plurality of candidate character strings produced by combining said plurality of recognition candidates employing the abovementioned cluster-code digram, the same effects as achieved with claim 16 are afforded.

[Brief Description of the Drawings]

[FIG. 1] is a function block diagram of a probability table generating apparatus of this invention;

[FIG. 2] is an explanatory diagram of a method for generating a compressed digram from a digram by means of a compressed digram generator, vector divider and clusterer of FIG. 1;

[FIG. 3] is a diagram of an example of attributes of Japanese language characters;

[FIG. 4] is a flow chart of the actuation of probability table generation processing executed by a controller of FIG. 1;

[FIG. 5] is a flow chart continuous to FIG. 4 of the actuation of the probability table generation processing;

[FIG. 6] is a diagram of memory size of a table generated by the probability table generating apparatus shown in FIG. 1;

[FIG. 7] is a function block diagram of a recognition apparatus of the invention;

[FIG. 8] is a flow chart of actuation of character recognition processing executed by a controller of FIG. 7; and

[FIG. 9] is a diagram showing a memory size of a conventional digram and trigram.

[Explanation of Symbols]

1 Document database, 12 Character attribute conversion table, 13 Compressed character code conversion table, 18, 46 Controller, 19 Unigram generator, 20 Digram generator, 21 Attribute trigram generator, 22 Compressed digram generator, 23 Vector divider, 24 Clusterer, 31 Image inputter, 32 Segmenter, 37 Feature extractor, 38 Matcher, 42 Candidate character string expander, 43 Score calculator, 45 Word inquiry language processing apparatus, 58 Table, 59 Word inquiry language dictionary.

[FIG. 9]

PROBABILITY TABLE	MEMORY SIZE	MEMORY SIZE CALCULATION EQUATION
DIGRAM	16 MEGABYTES	4000 X 4000
TRIGRAM	64 MEGABYTES	4000 X 4000 X 4000

[FIG. 1]

1 DOCUMENT DATABASE

2 UNIGRAM BUFFER

3 TOTAL CHARACTER NUMBER COUNTER

4 DIGRAM BUFFER

5 TOTAL 2-CHARACTER COMBINATION NUMBER COUNTER

6 ATTRIBUTE TRIGRAM BUFFER

7 TOTAL 3- CHARACTER COMBINATION NUMBER COUNTER

8 COMPRESSED DIGRAM BUFFER

9 TOTAL 2-COMPRESSED CHARACTER CODE COMBINATION NUMBER COUNTER

10 VECTOR BUFFER

11 CLUSTER BUFFER

12 CHARACTER ATTRIBUTE CONVERSION TABLE

13 COMPRESSED CHARACTER CODE CONVERSION TABLE

14 TEMPORARY MEMORY

15 CURRENT CHARACTER BUFFER

16 PREVIOUS CHARACTER BUFFER

17 PRE-PREVIOUS CHARACTER BUFFER

18 CONTROLLER

19 UNIGRAM GENERATOR

20 DIGRAM GENERATOR

21 ATTRIBUTE TRIGRAM GENERATOR

22 COMPRESSED DIGRAM GENERATOR

23 VECTOR DIVIDER

24 CLUSTERER

[FIG. 3]

ATTRIBUTE	CHARACTER TYPE	CHARACTER ATTRIBUTE CODE
ATTRIBUTE HI	HIRAGANA	0
ATTRIBUTE KA	KATAKANA	1
ATTRIBUTE SYM	SYMBOL	2
ATTRIBUTE KAN	KANJI	3
ATTRIBUTE NUM	NUMERAL	4
ATTRIBUTE UPP	UPPER CASE ALPHABET	5
ATTRIBUTE LOW	LOWER CASE ALPHABET	6

[FIG. 2]

[FIG. 6]

TABLE NAME	MEMORY SIZE	MEMORY SIZE CALCULATION EQUATION
UNIGRAM	4 KILOBYTES	4000 X 1
COMPRESSED DIGRAM	1 MEGABYTE	1000 X 1000 X 1
ATTRIBUTE TRGRAM	343 BYTES	7 X 7 X 7 X 1
CHARACTER ATTRIBUTE CONVERSION TABLE	4 KILOBYTES	4000 X 1
COMPRESSION CHARACTER CODE CONVERSION TABLE	8 KILOBYTES	4000 X 2

[FIG. 4]

START

S1 ALL BUFFERS AND COUNTERS CLEARED TO 0

S2 1-CHARACTER INPUT

S3 CHARACTER PRESENT?

S4 TOTAL CHARACTER NUMBER COUNT

S5 UNIGRAM COMPUTATION

S6 PREVIOUS CHARACTER PRESENT?

S7 TOTAL 2-CHARACTER COMBINATION NUMBER COUNT

S8 DIGRAM COMPUTATION

S9 PRE-PREVIOUS CHARACTER PRESENT?

S10 CHARACTER ATTRIBUTE CONVERSION

S11 TOTAL 3-CHARACTER ATTRIBUTE COMBINATION COUNT

S12 ATT TRIGRAM COMPUTATION

S13 PREVIOUS CHARACTERS STORED IN PRE-PREVIOUS CHARACTER BUFFER 16
CURRENT CHARACTERS STORED IN PREVIOUS CHARACTER BUFFFER 17

S14 UNIGRAM, DIGRAM, ATTRIBUTE TRIGRAM CONVERTED TO PROBABILITY VALUE

S15 UNIGRAM, DIGRAM, ATTRIBUTE TRIGRAM LOGARITHMICALLY COMPRESSED

[FIG. 5]

S16 DIGRAM COLUMN DECOMPOSED TO 4000 VECTOR

S17 4000 VECTOR CLUSTERED TO MAXIMUM 1000 CLUSTER

S18 COMPRESSED CHARACTER CODE CONVERSION TABLE 13 GENERATED

S19 COMPRESSED DIGRAM BUFFER 8, TOTAL 2-COMPRESSED CHARACTER CODE COMBINATION COUNTER 9, CURRENT CHARACTER BUFFER 15, PREVIOUS CHARACTER BUFFER 16 CLEARED TO 0

S20 1-CHARACTER INPUT

S21 CHARACTER PRESENT?

S22 PREVIOUS CHARACTER PRESENT?

S23 COMPRESSED CHARACTER CODE CONVERSION

S24 TOTAL 2-COMPRESSED CHARACTER CODE COMBINATION COUNT

S25 COMPRESSED DIGRAM COMPUTATION

S26 CURRENT CHARACTERS STORED IN PREVIOUS CHARACTER BUFFER 16

S27 COMPRESSED DIGRAM CONVERTED TO PROBABILITY TABLE

S28 COMPRESSED DIGRAM LOGARITHMICALLY COMPRESSED

END

[FIG. 7]

47 SCANNER

48 IMAGE FILER

49 IMAGE MEMORY

50 AREA COORDINATE MEMORY

51 COLUMN COORDINATE MEMORY

52 (TEMPORARY) CHARACTER COORDINATE MEMORY

53 PERIPHERAL DISTRIBUTION FEATURE BUFFER

54 FEATURE MEMORY

55 PATTERN DICTIONARY

56 RECOGNITION RESULT CANDIDATE BUFFER

57 CANDIDATE CHARACTER STRING BUFFER

58 TABLE · UNIGRAM
· COMPRESSED DIGRAM
· ATTRIBUTE TRIGRAM
· CHARACTER ATTRIBUTE CONVERSION TABLE
· COMPRESSED CODE CONVERSION TABLE

59 WORD INQUIRY LANGUAGE DICTIONARY

60 RECOGNITION RESULT BUFFER

31 IMAGE INPUTTER

32 SEGMENTER
33 AREA SEGMENTER
34 COLUMN SEGMENTER
35 CHARACTER SEGMENTER
36 TOUCH-CHARACTER SEGMENTER

37 FEATURE EXTRACTOR

38 MATCHER
39 FIRST SIMILARITY CALCULATOR
40 FIRST SORTER

41 CHARACTER STRING GENERATOR
42 CANDIDATE CHARACTER STRING EXPANDER
43 SCORE CALCULATOR
· SECOND SIMILARITY CALCULATOR
· PROBABILITY CALCULATOR

44 SECOND SORTER

45 WORD INQUIRY LANGUAGE PROCESSING APPARATUS

46 CONTROLLER

[FIG. 8]

START

S31 IMAGE INPUT

S32 AREA SEGMENTATION (OR AREA DESIGNATION)

S33 1-AREA IMAGE DATA READ

S34 AREA PRESENT?

S35 ALL COLUMNS SEGMENTED

S36 COLUMN PRESENT?

S37 1-COLUMN IMAGE DATA READ

S38 (TEMPORARY) CHARACTER SEGMENTATION

S39 CANDIDATE CHARACTER STRING BUFFER 57 INITIALIZATION

S40 CHARACTER PRESENT?

S41 1-CHARACTER SEGMENTATION

S42 FEATURE EXTRACTION

S43 MATCHING

S44 CHARACTER STRING EXPANSION

S45 SCORE CALCULATION
· SIMILARITY CALCULATION
· PROBABILITY CALCULATION

S46 SORTING

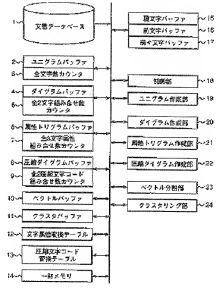
S47 WORD INQUIRY LANGUAGE PROCESSING

S48 STORAGE IN RECOGNITION RESULT BUFFER 60

S49 OUTPUT RECOGNITION RESULT

END

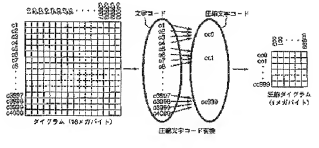
【図1】



【図3】

属性	文字種	文字属性コード
属0	非標点	0
属1	句読点	1
属2	記号	2
属3	漢字	3
属4	数字	4
属5	アルファベット大文字	5
属6	アルファベット小文字	6

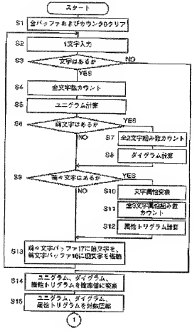
【図2】



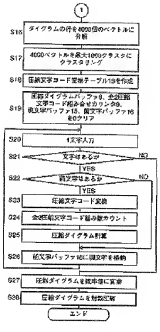
【図6】

テーブル名	メモリサイズ	メモリサイズ計算式
ユニグラム	4キロバイト	4096×1
圧縮ダイグラム	1メガバイト	N2[0K/1K]×1
属性トリグラム	3kBバイト	7×7×7×1
文字属性変換テーブル	4キロバイト	4096×1
圧縮文字コード変換テーブル	8キロバイト	4096×2

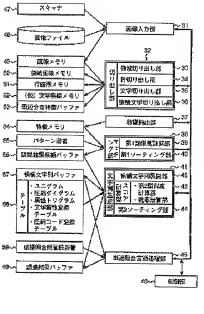
【図4】


スタート
S1 全バッファおよびカウンタクリア
S2 1文字入力
S3 文字はあるか
NO
YES
S4 全文字数カウント
S5 ユニグラム計算
S6 特文字はあるか
YES
NO
S7 特文字総合数カウント
S8 ダイグラム計算
S9 残り文字はあるか
YES
NO
S10 文字属性変換
S11 全文字属性総合数カウント
S12 属性トリグラム計算
S13 残り文字バッファIIIに後文字を、特文字バッファIVに固文字を転送
S14 ユニグラム、ダイグラム、属性トリグラムを標準値に変換
S15 ユニグラム、ダイグラム、属性トリグラムを対数圧縮
①

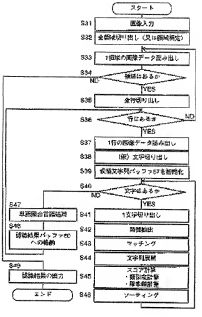
【図5】


①
S16 ダイグラムの許る4096個のベクトルに分類
S17 4096ベクトルを最大100のクラスタにクラスタリング
S18 圧縮文字コード変換テーブルIIIを作成
S19 英語ダイグラム、フラグ、全文書数、文字コード総合数サウンド、後、前文字バッファII、固文字バッファIVをゼロクリア
S20 1文字入力
S21 文字はあるか
NO
YES
S22 特文字はあるか
NO
YES
S23 圧縮文字コード変換
S24 全圧縮文字コード総合数カウント
S25 圧縮ダイグラム計算
S26 後文字バッファIIIに圧縮文字を移動
S27 圧縮ダイグラムを標準値に変換
S28 圧縮ダイグラムを対数圧縮
エンド

【図7】


47 スキャナ
48 画像ファイル
45 画像メモリ
52 領域情報メモリ
51 作業用メモリ
53 (他) 文字認識メモリ
53 罫線交差特徴バッファ
54 特徴メモリ
55 パターン辞書
56 認識結果候補バッファ
57 帳票全体用バッファ
58 テーブル
・ユニグラム
・見出しイクラム
・見出しトリグラム
・文字種別登録テーブル
・区切コード登録テーブル
59 認識結果出力書
59 認識用知識バッファ
60 処理部
61 画像入力部
62 領域切り出し部
罫線切り出し用
文字切り出し部
領域文字切り出し部
63 特徴抽出部
64 パターン照合部
65 第1認識処理部
第1ソーティング部
66 帳票全体用認識部
67 ユニグラム・見出しイクラム・見出しトリグラム・既知文書照合
68 第2ソーティング部
69 最適組合せ領域処理部
70 第2認識処理部

【図8】


スタート
S31 画像入力
S32 全領域切り出し（見出領域含む）
S33 1領域の属性データ読み出し
S34 領域はあるか
NO
YES
S35 全行読み出し
S36 行はあるか
NO
YES
S37 1行の画像データ読み出し
S38 (単) 文字切りばし
S39 仮想文字列バッファを初期化
S40 文字はあるか
NO
YES
S41 1文字切り出し
S42 特徴抽出
S43 マッチング
S44 文字列展開
スコア計算
類似度計算
降順並替え
S45 ソーティング
S46 文字はあるか
NO
YES
S47 単語照合言語処理
S48 領域結果バッファへの格納
S49 認識結果の出力
エンド